What You “Know” about Software and Safety is Probably Wrong

Prof. Nancy G. Leveson

MIT

© Nancy G. Leveson, 7/2020
Understanding The Problem

“It’s never what we don’t know that stops us. It’s what we do know that just ain’t so.”
General Definition of “Safety”

• **Accident = Mishap = Loss:** Any undesired and unplanned event that results in a loss
  – e.g., loss of human life or injury, property damage, environmental pollution, mission loss, negative business impact (damage to reputation, etc.), product launch delay, legal entanglements, etc. [MIL-STD-882]
  – Includes inadvertent and intentional losses (security)

• **System goals vs. constraints (limits on how can achieve the goals)**

• **Safety: Absence of losses**
Safety Engineering is about **Hazards**

- **Hazard**: A system state or set of conditions that, together with a particular set of worst-case environmental conditions, will lead to an accident.

- **Examples**:
  - Release of nuclear materials
  - Friendly fire
  - Loss of control of an aircraft
  - Violation of minimum separation between autos/planes

- Software is not unsafe. It can **contribute** to a hazard, but it does not explode, catch on fire, involve toxic materials, etc.

- If it is not about hazards, it is not about safety.
System Safety Overview

• A planned, disciplined, and systematic approach to preventing or reducing accidents throughout the life cycle of a system.

• Primary concern is the management of hazards

<table>
<thead>
<tr>
<th>Hazard</th>
<th>Through</th>
</tr>
</thead>
<tbody>
<tr>
<td>identification</td>
<td>analysis</td>
</tr>
<tr>
<td>evaluation</td>
<td>design</td>
</tr>
<tr>
<td>elimination</td>
<td>management</td>
</tr>
<tr>
<td>control</td>
<td></td>
</tr>
</tbody>
</table>

• Hazard analysis and control is a continuous, iterative process throughout system development and use.
It’s still hungry … and I’ve been stuffing worms into it all day.
A General Model of Control

- Software is not unsafe; the control signals it generates can be.
- Virtually all software-related accidents have resulted from unsafe requirements; not software design/implementation errors.
Our current tools are all 50-65 years old but our technology is very different today.

- Introduction of computer control
- Exponential increases in complexity
- New technology
- Changes in human roles

Assumes accidents caused by component failures

© Copyright Nancy Leveson, June 2011
It's only a random failure, sir! It will never happen again.
What Failed Here?

- Navy aircraft were ferrying missiles from one location to another.
- One pilot executed a planned test by aiming at aircraft in front and firing a dummy missile.
- Nobody involved knew that the software was designed to substitute a different missile if the one that was commanded to be fired was not in a good position.
- In this case, there was an antenna between the dummy missile and the target so the software decided to fire a live missile located in a different (better) position instead.
Accident with No Component Failures

• Mars Polar Lander
  – Have to slow down spacecraft to land safely
  – Use Martian atmosphere, parachute, descent engines (controlled by software)
  – Software knows landed because of sensitive sensors on landing legs. Cut off engines when determine have landed.
  – But “noise” (false signals) by sensors generated when landing legs extended. Not in software requirements.
  – Software not supposed to be operating at that time but software engineers decided to start early to even out the load on processor
  – Software thought spacecraft had landed and shut down descent engines while still 40 meters above surface
Warsaw A320 Accident

- Software protects against activating thrust reversers when airborne
- Hydroplaning and other factors made the software think the plane had not landed
- Pilots could not activate the thrust reversers and ran off end of runway into a small hill.
Boeing 787 Lithium Battery Fires

Certified based on models predicting 787 battery thermal problems would occur once in 10 million flight hours…but two batteries overheated in just two weeks in 2013
Boeing 787 Lithium Battery Fires

- A module monitors for smoke in the battery bay, controls fans and ducts to exhaust smoke overboard.
- Power unit monitors for low battery voltage, shut down various electronics, including ventilation.
- Smoke could not be redirected outside cabin.

All software requirements were satisfied! The requirements were unsafe.
Washington State Ferry Problem

- Local rental car company installed a security device to prevent theft by disabling cars if car moved when engine stopped
- When ferry moved and cars not running, disabled them.
- Rental cars could not be driven off ferries when got to port
Two Types of Accidents

- **Component Failure Accidents**
  - Single or multiple component failures
  - Usually assume random failure

- **Component Interaction Accidents**
  - Arise in interactions among components
  - Related to complexity (coupling) in our system designs, which leads to system design and system engineering errors
  - No components may have “failed”
  - Exacerbated by introduction of computers and software but the problem is system design errors
    - Software allows almost unlimited complexity in our designs
Confusing Safety and Reliability

Preventing Component or Functional Failures is Not Enough
Software Impact on Safety

1. Software allows almost unlimited system complexity

- Can no longer
  - Plan, understand, anticipate, and guard against all undesired system behavior
  - Exhaustively test to get out all design errors

- Context determines whether software is safe
  - Ariane 4 software was safe but when reused in Ariane 5, the spacecraft exploded
  - “SIL” (safety integrity level) concept is technically meaningless
  - “Level of Rigor” or “Design Assurance Level” (DAL) has nothing to do with the problem
  - Not possible to look at software alone and determine “safety”
Safe or Unsafe?
Safety Depends on Context
2. The role of software in accidents almost always involves flawed requirements

- Incomplete or wrong assumptions about operation of controlled system or required operation of computer
- Unhandled controlled-system states and environmental conditions

• Only trying to get the software “correct” or to make it reliable will not make it safer under these conditions
2. The role of software in accidents almost always involves flawed requirements

- Incomplete or wrong assumptions about operation of controlled system or required operation of computer
- Unhandled controlled-system states and environmental conditions

- Only trying to get the software “correct” or to make it reliable will not make it safer under these conditions
2. The role of software in accidents almost always involves flawed requirements

- Incomplete or wrong assumptions about operation of controlled system or required operation of computer
- Unhandled controlled-system states and environmental conditions

• Only trying to get the software “correct” or to make it reliable will not make it safer under these conditions
Software changes the role of humans in systems

Typical assumption is that operator error is cause of most incidents and accidents

– So do something about operator involved (admonish, fire, retrain them)

– Or do something about operators in general
  • Marginalize them by putting in more automation
  • Rigidify their work by creating more rules and procedures

“Cause” from the American Airlines B-757 accident report (in Cali, Columbia):

  “Failure of the flight crew to revert to basic radio navigation at the time when the FMS-assisted navigation became confusing and demanded an excessive workload in a critical phase of flight.”
Fumbling for his recline button Ted unwittingly instigates a disaster
Another Accident Involving Thrust Reversers

- Tu-204, Moscow, 2012
- Red Wings Airlines Flight 9268
- The soft 1.12g touchdown made runway contact a little later than usual.
- With the crosswind, this meant weight-on-wheels switches did not activate and the thrust-reverse system would not deploy.
Another Accident Involving Thrust Reversers

- Pilots believe the thrust reversers are deploying like they always do. With the limited runway space, they quickly engage high engine power to stop quicker. Instead this accelerated the Tu-204 forwards, eventually colliding with a highway embankment.
Another Accident Involving Thrust Reversers

• Pilots believe the thrust reversers are deploying like they always do. With the limited runway space, they quickly engage high engine power to stop quicker. Instead this accelerates the Tu-204 forwards, eventually colliding with a highway embankment.

In complex systems, human and technical considerations cannot be isolated
Human factors concentrates on the “screen out”

Hardware/Software engineering concentrates on the “screen in”
Not enough attention on integrated system as a whole

(e.g., mode confusion, situation awareness errors, inconsistent behavior, etc.)
The New Systems View of Operator Error

• Operator error is a symptom, not a cause

• All behavior affected by context (system) in which occurs
  – Role of operators is changing in software-intensive systems as is the errors they make
  – Designing systems in which operator error inevitable and then blame accidents on operators rather than designers

• To do something about operator error, must look at system in which people work:
  – Design of equipment
  – Usefulness of procedures
  – Existence of goal conflicts and production pressures

• **Human error is a symptom of a system that needs to be redesigned**
Summary of the Problem:

- We need models and tools that handle:
  - Hardware and hardware failures
  - Software (particularly requirements)
  - Human factors
  - Interactions among system components
  - System design errors
  - Management, regulation, policy
  - Environmental factors
  - “Unknown unknowns”

And the interactions among all these things
It’s still hungry … and I’ve been stuffing worms into it all day.
It’s still hungry … and I’ve been stuffing worms into it all day.

We Need New Tools for the New Problems
The Problem is Complexity

Ways to Cope with Complexity

• Analytic Decomposition

• Statistics

• Systems Theory
Analytic Decomposition ("Divide and Conquer")

1. Divide system into separate parts

*Physical/Functional: Separate into distinct components*

- Components interact in direct ways
- Each event is the direct result of the preceding event

*Behavior: Separate into events over time*
2. Analyze/examine pieces separately and combine results

- Assumes such separation does not distort phenomenon
  - Each component or subsystem operates independently
  - Components act the same when examined singly as when playing their part in the whole
  - Components/events not subject to feedback loops and non-linear interactions
  - Interactions can be examined pairwise
Bottom Line

• These assumptions are no longer true in our
  – Tightly coupled
  – Software intensive
  – Highly automated
  – Connected
  engineered systems

• Need a new theoretical basis
  – *System theory* can provide it
Degree of Randomness

Degree of Coupling

Unorganized Complexity
(can use statistics)

Organized Complexity

Organized Simplicity
(can use analytic decomposition)

[Credit to Gerald Weinberg]
Here comes a paradigm change for safety and security!

Safety as a **Failure** Problem  

Safety as a **Control** Problem
Systems Theory

• Developed for systems that are
  – Too complex for complete analysis
    • Separation into (interacting) subsystems distorts the results
    • The most important properties are emergent
  – Too organized for statistics
    • Too much underlying structure that distorts the statistics
    • New technology and designs have no historical information

• First used on ICBM systems of 1950s/1960s

System Theory was created to provide a more powerful way to deal with complexity
Systems Theory (2)

• Focuses on systems taken as a whole, not on parts taken separately

• Emergent properties
  – Some properties can only be treated adequately in their entirety, taking into account all social and technical aspects
    “The whole is greater than the sum of the parts”
  – These properties arise from relationships among the parts of the system
    How they interact and fit together
Emergent properties
(ariise from complex interactions)

The whole is greater than
the sum of its parts

Process components interact in
direct and indirect ways
Emergent properties
(arise from complex interactions)

The whole is greater than the sum of its parts

Process components interact in direct and indirect ways

Safety and security are emergent properties
Controller

Controlling emergent properties (e.g., enforcing safety constraints)
- Individual component behavior
- Component interactions

Control Actions → Process

Feedback → Process

Process components interact in direct and indirect ways
Controls/Controllers Enforce Safety Constraints

- Power must never be on when access door open
- Two aircraft/automobiles must not violate minimum separation
- Aircraft must maintain sufficient lift to remain airborne
- Integrity of hull must be maintained on a submarine
- Toxic chemicals/radiation must not be released from plant
- Workers must not be exposed to workplace hazards
- Public health system must prevent exposure of public to contaminated water and food products
- Pressure in a offshore well must be controlled

These are the High-Level Functional Hazard-Related Safety/Security Requirements to Address During Design
Controller

- Controlling emergent properties (e.g., enforcing safety constraints)
  - Individual component behavior
  - Component interactions

Air Traffic Control: Safety Throughput

Process components interact in direct and indirect ways
Treat the Software/Humans as a Feedback Control System

- Controllers use a **process model** to determine control actions
- Software/human related accidents often occur when the process model is incorrect
- Captures software errors, human errors, flawed requirements ...

![Feedback Control System Diagram]

- **Controller**
  - Control Algorithm
  - Process Model

- **Controlled Process**
- **Control Actions** (via actuators)
- **Feedback** (via sensors)
Mars Polar Lander

**Hazard:** landing on planet with too much force

**Spacecraft Software**
- Control Algorithm
- Process Model

**Spacecraft**

- Turn off descent engines (via actuators)
- Feedback (via sensors)
- Land leg sensor feedback
- Spacecraft has landed
Unsafe Control Actions

Four types of unsafe control actions

1) Control commands required for safety are not given
2) Unsafe commands are given
3) Potentially safe commands but given too early, too late
4) Control action stops too soon or applied too long (continuous control)

Analysis:
1. Identify potential unsafe control actions
2. Identify why they might be given
3. If safe ones provided, then why not followed?
Figure 3.9: System block diagram. A is the primary and B is the redundant system.
High-level control structure

Ground Station

Spacecraft

Attitude and Orbit Control System (AOCS)

- Attitude Control System (ACS)
- Reaction Control System (RCS)

Science Instruments

- Soft X-Ray (SX)
- Hard X-Ray (HX)
- Soft gamma ray (SG)
Spacecraft

Attitude and Orbit Control System (AOCS)
- Attitude Control System (ACS)
- Reaction Control System (RCS)

Ground Station

Software-hardware interactions

Controller
- Control Algorithm
- Process Model

Controlled Process

Feedback

Control Actions
High-level control structure

- Ground Station
- Spacecraft
  - Attitude and Orbit Control System (AOCS)
    - Attitude Control System (ACS)
  - Reaction Control System (RCS)
  - Human-Automation interactions

Controller
  - Control Algorithm
  - Process Model

Controlled Process

Feedback

Control Actions
High-level control structure

Ground Station

Science Instruments

- Soft X-Ray (SX)
- Hard X-Ray (HX)
- Soft gamma ray (SG)

Human-hardware interactions

Controller

- Control Algorithm
- Process Model

Controlled Process

Feedback

Control Actions

Spacecraft
Integrated Approach to Safety and Security

• Both concerned with losses (intentional or unintentional)
  – Mission assurance (vs. information protection)
  – Ensure that critical functions and services are maintained
  – New paradigm for safety will work for security too
    • May have to add new causes, but rest of process is the same
  – A top-down, system engineering approach to designing safety and security into systems
Example: Stuxnet

- **Loss**: Damage to reactor (in this case centrifuges)
- **Hazard/Vulnerability**: Centrifuges are damaged by spinning too fast
- **Constraint to be Enforced**: Centrifuges must never spin above maximum speed
- **Hazardous control action**: Issuing *increase speed* command when already spinning at maximum speed
- One potential causal scenario:
  - Incorrect process model: thinks spinning at less than maximum speed
    - Could be inadvertent or deliberate
- **Potential controls**:
  - Mechanical limiters (interlock), Analog RPM gauge

Focus on preventing hazardous state (not keeping intruders out)
STAMP
(System-Theoretic Accident Model and Processes)

- A new, more powerful accident/loss causality model
- Based on systems theory, not reliability theory
- Defines accidents/losses as a dynamic control problem (vs. a failure problem)
- Applies to VERY complex systems
- Includes
  - Scenarios from traditional hazard analysis methods (failure events)
  - Component interaction accidents
  - Software and system design errors
  - Human errors
  - Entire socio-technical system (not just technical part)
STAMP: Theoretical Causality Model
How is it being used?
Does it work?
Is it useful?
Is it Practical?

• STPA has been or is being used in a large variety of industries
  – Automobiles (>80% use)
  – Aircraft and UAVs (extensive use and growing)
  – Defense systems
  – Air Traffic Control
  – Spacecraft
  – Medical Devices and Hospital Safety
  – Chemical plants
  – Oil and Gas
  – Nuclear and Electric Power
  – Robotic Manufacturing / Workplace Safety
  – Pharmaceuticals
  – etc.

• New international standards for STPA or in development
Evaluations and Estimates of ROI

• Hundreds of evaluations and comparison with traditional approaches used now
  – Controlled scientific and empirical (in industry)
  – All show STPA is better (identifies more critical requirements or design flaws)
  – All (that measured) show STPA requires orders of magnitude fewer resources than traditional techniques
  – Successfully finds accidents before they occur

• ROI estimates only beginning but one large defense industry contractor claims they are seeing 15-20% return on investment when using STPA
To Make Progress We Need to:

• Develop and use different approaches that match the world of engineering today and the problems today

• Consider the entire sociotechnical system

• Focus on building safety/security in rather than assuring/measuring it after the design is completed

“The best way to predict the future is to create it.”

Abraham Lincoln

Start from the problem, not a solution (avoid feeding worms to the baby)
More Information

• [http://psas.scripts.mit.edu](http://psas.scripts.mit.edu) (papers, presentations from conferences, tutorial slides, examples, etc.)

Free download: [http://mitpress.mit.edu/books/engineering-safer-world](http://mitpress.mit.edu/books/engineering-safer-world)

STAMP Virtual Workshop

July 20 to Aug. 7

Tutorials, industry presentations, research presentations

PSAS website: http://psas.scripts.mit.edu
Standard Safety Approach does not Handle

- Component interaction accidents
- Systemic factors (affecting all components and barriers)
- Software and software requirements errors
- Human behavior (in a non-superficial way)
- System design errors
- Indirect or non-linear interactions and complexity
- Culture and management
- Migration of systems toward greater risk over time (e.g., in search for greater efficiency and productivity)
• Old Assumption: accidents are caused by component failure
• New Reality: Accidents involving software usually do not involve component failure.
• Accidents are not the result of random failure or even errors/faults.
A Broad View of “Control”

Component failures and unsafe interactions may be “controlled” through design
  (e.g., redundancy, interlocks, fail-safe design)

or through process
  – Manufacturing processes and procedures
  – Maintenance processes
  – Operations

or through social controls
  – Governmental or regulatory
  – Culture
  – Insurance
  – Law and the courts
  – Individual self-interest (incentive structure)
Warsaw (Reverse Thrusters)

**Hazard:** Inadequate a/c deceleration after landing

- **Pilot**
  - Decision Making
  - Process Model

- **Software Controller**
  - Control Algorithm
  - Process Model

- **Aircraft**

  - Turn on reverse thrusters
  - Ignore command

- Plane has landed
- Plane has not landed
- Feedback indicates plane has not landed
Moscow (Reverse Thrusters)

Hazard: Inadequate Deceleration after Landing

Pilot
- Decision Making
- Process Model

Software Controller
- Control Algorithm
- Process Model

Aircraft

Plane has landed
Plane has not landed
Feedback indicates plane has not landed
Ignore reverse thruster command
**Moscow (Reverse Thrusters)**

**Hazard: Inadequate Deceleration after Landing**

- **Pilot**
  - Decision Making
  - Process Model

- **Software Controller**
  - Control Algorithm
  - Process Model

**Aircraft**

- **Plane has landed**
  - Reverse thrusters will come on

- **Plane has not landed**
  - Feedback indicates plane has not landed

- **Ignore reverse thruster command**

- **Engage high engine power**

- **Engage reverse thrust**

- **Engage high engine power**

- **Short runway, need more power to stop**
Missile Release Mishap

Hazard: Friendly Fire

- Perform test with dummy missile
- Launch dummy missile
- Optimize missile success
- Launch live missile
- Live missile in better position to hit target
Boeing Lithium-ion Batteries

Hazard: Fire and Smoke

- Special investigation of
  - Multiple controllers of same process
  - Boundaries between processes with multiple controllers
Unsafe Control Actions

Four types of unsafe control actions

1) Control commands required for safety are not given
2) Unsafe commands are given
3) Potentially safe commands but given too early, too late
4) Control action stops too soon or applied too long (continuous control)

Analysis:
1. Identify potential unsafe control actions
2. Identify why they might be given
3. If safe ones provided, then why not followed?
A NEW MODEL FOR HUMAN CONTROLLERS

Captures the controller’s goals and how decisions are made based on the mental models

Captures specific types of flaws in the way the human controller conceptualizes the system and environment

Captures the influence of human experiences, and expectations on the processing of sensory input

Provides an alternative to the existing controller model which is better suited for software controllers

(Thomas & France, 2016)
Safety as a Dynamic Control Problem (STAMP)

- Hazards result from lack of enforcement of safety constraints in system design and operations.

- Goal is to control the behavior of the components and systems as a whole to ensure safety constraints are enforced in the operating system.

- A change in emphasis:
  
  *Increase component reliability (prevent failures)*

  *Enforce safety/security constraints on system behavior*

  (note that enforcing constraints might require preventing failures or handling them but includes more than that)